Projet d'IOT

HAI912I

M2 Informatique - Génie Logiciel

Faculté des Sciences de Montpellier

TRINQUART Matthieu

SANCHEZ Martin

15 janvier 2023

Sommaire

1 Introduction

Ce projet s'inscrit dans le cadre de notre formation pour la matière HAI912I, Développement mobile avancé, IOT et embarqué. Nous avons décidé de faire équipe pour ce projet, car nous partageons un intérêt particulier pour l'automatisation de certaines tâches domotiques.

L'objectif du projet est d'implémenter une API REST sur une carte ESP32, en effet la carte dispose d'un module Wifi qui peut être utilisé comme récepteur pour se connecter à un réseau, comme émetteur pour créer un réseau, ou les deux.

Cette API doit permettre de consulter les valeurs des capteurs connectés à l'ESP32, dans notre cas nous avons une thermo-résistance et une photo-résistance. Elle doit aussi permettre de contrôler l'allumage, l'extinction, la mise en mode automatique de la LED. Ce mode automatique, permettra d'allumer la LED seulement en dessous d'un seuil de luminosité fixé.

La carte dispose d'un petit écran, nous nous en servirons pour afficher les valeurs de luminosité et de température.

2 Notre projet

2.1 Organisation

Notre organisation dans ce projet a été dans un premier temps de se séparer les tâches. En effet nous avons pensé que le web service et la partie capteurs pouvaient être réalisé séparément.

Nous nous sommes donc partagés ces tâches, cela nous a permis d'avancer en parallèle. Une fois les deux parties terminées il nous a suffit de faire correspondre les routes du service web aux variables et procédures correspondant au fonctionnement des capteurs.

2.2 Le montage

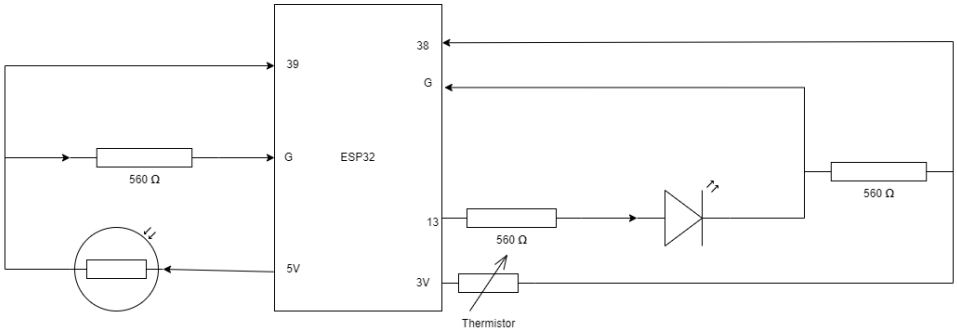

FIGURE 1 – Schéma électrique du montage

Pour le montage nous avons connecté la photo-résistance sur la broche de 5V, le circuit se sépare ensuite en deux parties.

— Une partie est reliée à la terre avec une résistance de 560 ohm.
— L'autre est reliée à la broche 38 afin de mesurer la variation du courant.

Nous avons ensuite une LED sur la broche 13 avec une résistance de 560 ohm qui sera ensuite reliée à la terre.

Et pour finir, nous avons sur la broche 3V la thermo-résistance, la suite du circuit se sépare aussi en deux.

— Une partie connectée à une résistance de 560 ohm qui sera connecté à la terre.
— L'autre reliée à la broche 39 afin de mesurer la variation de courant de la thermo-résistance.

2.3 Gestion des entrées / sorties

L'objectif de cette partie était de traiter les données récupérées grâce aux photo- et thermo- résistances, et de convertir ces valeurs dans des unités plus intelligibles pour l'homme, nous avons choisi les degrés et les lux.
Nous afficherons ces informations sur l'écran du TTGO. Cela permettra de visualiser ces informations sans avoir besoin du service web.

Nous voulons aussi gérer l'allumage d'une LED selon 3 modes :
— Un mode allumé, la LED s'éclaire quoi qu'il arrive.
— Un mode éteint, la LED ne s'éclaire pas.
— Un mode automatique, la LED s'éclaire seulement si la luminosité donnée par le capteur est inférieure à 1500 lux.

Nous utilisons un certain nombre de constante pour nos conversions.

```
1     // Tension de la broche initiale
2 const double VCC = 3.3;
3     // En ohm, la résistance associée à la photo-résistance
4 const double R2 = 560;
5 const double adc_resolution = 1023;
6
7
8     // Constantes qui permettent de calculer la température
9 const double A = 0.001129148;
10 const double B = 0.000234125;
11 const double C = 0.0000000876741;
12
13    // Tension de la broche associée à la thermo-résistance
14 #define VIN 5
15     // En ohm, la résistance associée à la thermo-résistance
16 #define R 0.56
```

La température

Pour convertir la valeur de tension en degrés, voici notre procédure :

```
1       // Lecture de la tension
2  int analgtemp = analogRead(38);
3  Serial.println(analgtemp);
4      // Conversion
5  Vout = (analgtemp * VCC) ;
6  Rth = (VCC * R2 / Vout) - R2;
7  temperature = (1 / (A + (B * log(Rth)) + (C * pow((log(Rth)),3))));
8
9  temperature = temperature - 273.15;
```

Nous prenons la valeur de tension de la broche 38 où se situe la thermo-résistance. Nous affichons cette valeur et nous utilisons ensuite la formule suivante trouvé sur internet pour convertir en degrés la valeur de tension :

$$\text{Rth} = (\text{VCC} * \text{R2} / \text{Vout}) - \text{R2};$$
$$\text{temperature} = (1 / (\text{A} + (\text{B} * \log(\text{Rth})) + (\text{C} * \text{pow}((\log(\text{Rth})),3))));$$
$$\text{temperature} = \text{temperature} - 273.15;$$

VCC : correspond au voltage de base où est branchée la thermo-résistance dans notre cas 3V.
R2 : correspond en ohm à la valeur de la résistance associée à la thermo-résistance dans notre cas 560 ohm.
A, B et C sont des constantes trouvées sur internet qui permettent de transformer la valeur de la résistance en degrés kelvin.

Nous soustrayons 273.5 à cette valeur pour la convertir en degrés.

La lumière

Pour convertir la valeur de tension en Lux nous utilisons la fonction suivante trouvée sur internet :

```
int sensorRawToPhys(int raw){
  // Conversion rule
  float Vout = float(raw) * (VIN / float(1024));// Conversion analog to
   voltage
  float RLDR = (R * (VIN - Vout))/Vout; // Conversion voltage to resistance
  int phys=500/(RLDR/1000); // Conversion resitance to lumen
  return phys;
}
```

Elle prend en paramètre la valeur de tension en Volt reçue par la broche 39.
VIN correspond à la valeur de tension initiale (dans notre cas 5V).
R correspond à la valeur de la résistance associée à la photo-résistance dans notre cas 560 ohm.

Cette fonction va retourner la valeur en Lux de la lumière détectée par la photo-résistance.

L'écran

On affiche le résultats de la valeur de luminosité en Lux et la valeur de température en °C sur l'écran du TTGO.
Pour afficher du texte sur l'écran du TTGO nous utilisons les librairies suivantes :

```
#include <TFT_eSPI.h>
#include <SPI.h>
```

Nous utilisons la fonction TFT_eSPI tft = TFT_eSPI(); pour manipuler l'écran du TTGO.

Et le programme suivant pour y afficher les valeurs :

```
    // Nous placons le curseur sur l'écran.
tft.setCursor(0, 50, 2);
    // Nous définissons la couleur en noir.
tft.setTextColor(TFT_WHITE,TFT_BLACK);
    // Nous définissons la taille du texte.
tft.setTextSize(1);
    // On affiche la variable de température.
tft.print(temperature);
```

La couleur du fond de l'écran change aléatoirement à chaque démarrage du programme grâce à l'instruction :

```
tft.fillScreen(random(0xFFFF));
```

La LED

Pour allumer la LED nous utilisons le code suivant :

```
if(mode == 0){
    // La LED est allumée
    digitalWrite(13, HIGH);
}else if(mode ==1){
     // La LED est éteinte
    digitalWrite(13, LOW);
} else{
    // Le mode automatique
    if(lux > 1500){ led = false;
    }else{          led = true; }
}
```

La variable mode correspond à l'état voulu de la LED :

- mode = 0 veut dire que la LED sera allumée
- mode = 1 la LED sera éteinte
- mode = 2 la LED sera en mode automatique, c'est-à-dire allumée si la valeur de luminosité est inférieure à 1500 et éteinte sinon.

La LED sera connectée sur la broche 13.

2.4 Le web service

Nous avons choisi d'utiliser le module Wifi de la carte en mode émetteur, en effet nous créons notre propre réseau. Ceci nous permet de travailler hors du réseau de la fac, car il utilise une interface web pour la connexion, ce qui n'est pas faisable avec l'ESP.

Nous utilisons ces librairies, elles permettent d'avoir accès au module Wifi de la carte et de gérer un serveur http, nécessaire pour le web service. Le fichier "html.h" est un fichier qui contient le code HTML de la page d'accueil du web service.

```
#include <Arduino.h>
#include <WiFi.h>
#include <WebServer.h>
#include <FreeRTOS.h>

#include "esp_wifi.h"
#include "html.h"
```

Par la suite, nous créons un certain nombre de variables pour paramétrer le point d'accès Wifi.

```
const char* ssid  = "ESP_32_MM";
const char* password = "";
const int channel = 10;
const bool hide_SSID = false;
const int max_connection = 2;

WebServer server(80);
```

Ce code permet simplement de dire à l'ESP que le nom du réseau est "ESP_32_MM", nous ne définissons pas de mot de passe, cela permet de se connecter facilement au réseau. Pour nos tests c'était intéressant mais pour un usage réel il faudrait ajouter un mot de passe.
Nous autorisons 2 personnes connectées en simultané.
Nous ouvrons le serveur sur le port 80, port http standard.

```
void initWifi() {
    Serial.print("\n SSID: ");
    Serial.print(ssid) ;
    Serial.print("\n PSW: ");
    Serial.println(password) ;

    WiFi.mode(WIFI_AP);
    WiFi.softAP(ssid, password);

    Serial.println("Point d'accès ouvert !") ;
    Serial.println(WiFi.softAPIP());
}
```

Nous avons ensuite une fonction pour initialiser le réseau. On retrouve ici le choix de créer un point d'accès Wifi.
L'instruction : WiFi.mode(WIFI_AP) veut dire que nous utilisons l'ESP en mode émetteur.

```
void printConnectedUsers(){
    wifi_sta_list_t wifi_sta_list;
    tcpip_adapter_sta_list_t adapter_sta_list;
    esp_wifi_ap_get_sta_list(&wifi_sta_list);
    tcpip_adapter_get_sta_list(&wifi_sta_list, &adapter_sta_list);

    if (adapter_sta_list.num > 0)
        Serial.println("----------");
    for (uint8_t i = 0; i < adapter_sta_list.num; i++){
        tcpip_adapter_sta_info_t station = adapter_sta_list.sta[i];
        Serial.print((String)"[+] Device " + i + " | MAC : ");
        Serial.printf("%02X:%02X:%02X:%02X:%02X:%02X", station.mac[0],
    station.mac[1], station.mac[2], station.mac[3], station.mac[4], station.
    mac[5]);
        Serial.println((String) " | IP " + ip4addr_ntoa(&(station.ip)));
    }}
```

Nous avons aussi cette petite fonction, qui dans l'absolu n'est pas utile, mais qui nous permet de voir dans le moniteur de série la liste des appareils connectés à notre réseau. Cela nous a permis de voir que notre réseau était bien fonctionnel. Ce morceau de code nous l'avons trouvé sur un site uPesy.fr.

```
void setup_routing() {
  server.on("/", serverRoot);
  server.on("/readTemperature", readTemp);
  server.on("/readLumiere", readLum);
  server.on("/led/on", statusLedOn);
  server.on("/led/off", statusLedOff);
  server.on("/led/auto", statusLedAuto);

  // start server
  server.begin();
}
```

Nous avons ensuite cette fonction qui nous permet de gérer le routing du serveur, c'est à dire, quels chemins mènent à quelle fonctionnalité. Nous avons donc le chemin "/" qui est la racine du serveur, ce n'est pas particulièrement utile mais cela nous permet d'afficher une page web, donc au format HTML. Elle contiendra simplement des liens vers les autres routes dans le but de faciliter la navigation dans le service web.

Chacune de ces routes sont associés a des fonctions, ces fonctions ont pour mission de gérer la réponse du serveur.

```
void serverRoot() {
  Serial.println("racine du webservice");
  server.send(200, "text/html", HTML);
}
void readTemp() {
  Serial.println("/readTemperature = X");
  server.send(200, "application/json", {"Temperature = " + String(
   temperature)});
}
void readLum() {
  Serial.println("/readLumiere = X");
  server.send(200, "application/json", {"Lumiere = "+ String(lux)});
}
void statusLedOn() {
  Serial.println("/led = ON");
  server.send(200, "application/json", {"Led = On"});
}
void statusLedOff() {
  Serial.println("/led = OFF");
```

```
19    server.send(200, "application/json", {"Led = Off"});
20  }
21  void statusLedAuto() {
22    Serial.println("/led = AUTO");
23    server.send(200, "application/json", {"Led = Auto"});
24  }
```

Ces fonctions sont toutes simples, elles permettent de laisser une trace dans la console, et de renvoyer au client le code d'erreur 200, qui signifie que tout s'est bien passé. La fonction serverRoot() est la seule qui renvoie une page web, les autres renvoient du JSON. Les variables température et lux sont les variables qui contiennent les valeurs des capteurs, après conversion.

```
1  void setup(){
2      Serial.begin(115200);
3      initWifi();
4      setup_routing();
5  }
6  void loop() {
7      printConnectedUsers();
8      server.handleClient();
9  }
```

Et pour finir, il nous suffit d'appeler dans le setup l'initialisation du réseau et du routing. Ett dans la loop, nous affichons les utilisateurs qui sont connectés et nous demandons au serveur de traiter un client.

Pour cette partie, nous avons fait beaucoup de recherches sur internet, et nous avons trouvé deux tutoriels dont nous nous sommes inspirés pour écrire notre code. Un deux provient du site upesy, c'est un site qui vend du matériel électronique et qui propose un certain nombre de tutoriels.
Nous avons aussi trouvé un créateur de contenu, SurvivingWithAndroid, qui propose des tutoriels pour Arduino, Raspery PI, et ESP32.

Tutoriel uPesy.fr

Surviving with android

3 Résultats

Nous vous avons préparé une courte vidéo montrant le fonctionnement de notre travail. On y voit le service web qui lis les valeurs des capteurs et qui modifie l'état de la LED. Nous voyons aussi le fonctionnement du mode automatique de la LED, qui l'allume en dessous d'un seuil de lumière.

La vidéo de présentation sur youtube.

4 Conclusion

Pour conclure, ce projet était pour nous une excellente occasion de découvrir et d'obtenir certaines compétences de base dans ce domaine. Compétences que nous pourrons exploiter dans nos projet personnels.

Nous aurions pu améliorer certains points, comme par exemple le fait de pouvoir définir le seul de luminosité pour allumer la LED, cela nous aurai permis de découvrir comment gérer les requêtes POST sur le web service.

Nous n'avons pas non plus listé les différents capteurs branchés sur la carte.